

MDS Codes with Progressive Engagement Property for Cloud Storage Systems

Mahdi Hajiaghayi* and Hamid Jafarkhani*

*Center of Pervasive Communications and Computing, University of California, Irvine

Email: {mahdi.h, hamidj}@uci.edu

Abstract—Fast and efficient failure recovery is a new challenge for cloud storage systems with a large number of storage nodes. A pivotal recovery metric upon the failure of a storage node is *repair bandwidth cost* which refers to the amount of data that must be downloaded for regenerating the lost data. Since all the surviving nodes are not always accessible, we intend to introduce a class of maximum distance separable (MDS) codes that can be re-used when the number of selected nodes varies yet yields close to optimal repair bandwidth. Such codes provide flexibility in engaging more surviving nodes in favor of reducing the repair bandwidth without redesigning the code structure and changing the content of the existing nodes. We call this property of MDS codes *progressive engagement*. This name comes from the fact that if a failure occurs, it is shown that the best strategy is to incrementally engage the surviving nodes according to their accessing cost (delay, number of hops, traffic load or availability in general) until the repair-bandwidth or accessing cost constraints are met. We argue that the existing MDS codes fail to satisfy the progressive engagement property. We subsequently present a search algorithm to find a new set of codes named *rotation codes* that has both progressive engagement and MDS properties. Furthermore, we illustrate how the existing permutation codes can provide progressive engagement by modifying the original recovery scheme. Simulation results are presented to compare the repair bandwidth performance of such codes when the number of participating nodes varies as well as their speed of single failure recovery.

I. INTRODUCTION

With the advent of the cloud storage systems, ordinary users and enterprises are increasingly moving their data to the cloud in order to have higher reliability, availability, and accessibility. These massive data are distributed across a large number of storage nodes via a cloud file system and preserved for long time, perhaps forever. Due to various incidents, ranging from regional disasters such as earthquake, to hardware/software update on storage nodes, the cloud file system must be able to reconstruct the user data only from a subset of storage nodes. Such capability can be obtained through an efficient fault-tolerant system based on replication and redundancy.

Mirroring is a basic yet popular solution that is usually adopted by local storage systems. This solution which is also reflected in levels 1 – 2 of Redundant Array of Inexpensive Disk (RAID) [1] simply keeps an exact copy of the data in multiple, usually three, storage nodes. Mirroring results in high availability but it comes at the price of high redundancy.

While mirroring may work for a local storage system, its hardware and operation cost (power, cooling systems and

maintenance) drastically increases with the size of data, thus not efficient for a cloud systems. Therefore, cloud file systems are transitioning to adopt erasure codes [2], e.g. Windows Azure Storage [3] and Google GFS [4]. Maximum distance separable (MDS) codes [5] are optimal among erasure codes in terms of redundancy-reliability trade-off. An (n, k) MDS code when deployed into storage systems can tolerate up to $(n - k)$ node failures without data loss while requiring a much lower redundancy compared with mirroring. For an (n, k) MDS code to be applied to a symmetric storage system, we must divide a data of size M into k blocks and encode them into n equal-size encoded blocks and store them at n storage nodes. In this case, the whole data can be reconstructed from any k nodes out of the total n nodes. Coding in this context consists of two parts: First, encoding that determines the content of each node and second, decoding that specifies the recovery scheme when a node failure happens. Obviously, the decoding process highly depends on the encoding part. Several MDS codes such as Reed-Solomon [6], row-diagonal parity (RDP) [7], EVENODD [8], X-code [9], and WEAVER codes [10] were proposed and applied to the storage systems in the literature with specific features and decoding complexity to protect data against multiple disk failures.

Other codes such as low-density parity-check (LDPC) codes and related codes such as LT codes [11] and Raptor codes [12] have also been considered for distributed storage systems [13]. It is shown that while these codes exhibit a considerably low decoding complexity due to their XOR-based structure, they fail to hold the MDS property. To address this problem, [14] and [15] attempt to find some XOR-based MDS codes using an exhaustive search approach over all possible cases. Due to the exponential growing search space, they could only provide such codes for limited cases ($k < 20$, and $(n - k) < 10$). The authors of [16] aim at minimizing the number of nodes to be contacted for recovery while offering MDS property for the codes with rate less than $1/2$. Locally repairable codes (LRC) [17] also considers the same metric as the code design metric.

The amount of data that must be downloaded from d surviving nodes to repair a single failure is called *repair bandwidth* [18]. When a permanent failure occurs, the lost data must be recovered and regenerated at a new node to maintain the data integrity. The downside of the traditional MDS codes is that the entire data must be downloaded before recovering the lost data. Hence, its repair bandwidth is M , albeit to only M/k portion of the original data being lost in a failed node. This high repair bandwidth incurs considerable

file transfer and traffic on the network. This has sparked interest in finding the minimum repair bandwidth which in many cases has led to a new class of MDS codes [18],[19], [20], [21],[22] [23]. Such codes have been proposed for some specific combinations of (n, k, d) where d is the number of surviving nodes participating in the recovery. All these newly proposed MDS codes require less repair bandwidth than M and this bandwidth decreases as more nodes are engaged for recovery. However, both encoding and decoding parts of these codes depend on d and it must be known before designing the code. Therefore, for these MDS codes the coding structure and basically the content of nodes change as d changes. This is not practically appealing for two reasons. First, we may not know in advance how many nodes are available to participate for the recovery. Second, when a new node is added to the storage system, the content of the existing nodes must be changed for the optimal repair bandwidth performance.

In this paper, we intend to introduce a family of MDS codes that while requiring low repair bandwidth it can be reused when the number of participating nodes varies without having to change the entire code structure and the content of the nodes. This property which we call *progressive engagement* provides flexibility in engaging more surviving nodes in favor of reducing the repair bandwidth without redesigning the code structure. This name stems from a proven fact that if a failure occurs, the best strategy to recover the lost data in the presence of node access cost is to incrementally engage the surviving nodes according to their accessing cost (delay, number of hops, traffic load and availability in general) until the repair bandwidth or accessing cost constraints are met. Progressive engagement property has another implication that allows for adding new storage nodes to the system without changing the contents of the existing nodes.

A similar approach to the progressive engagement can be seen in rate-compatible punctured codes [24] for communication systems. Such codes allow the transmission of information with different levels of protection, at various rates without redesigning the code.

We argue that the existing MDS codes fail to satisfy this pivotal property. In traditional MDS codes such as Reed-Solomon codes, engaging more nodes does not lead to the reduction of repair bandwidth. On the other hand, progressive engagement is not satisfied in the existing MDS codes designed for reducing repair bandwidth [25], [20], [23] since their structure vary with the number of participating nodes. This fact renders them non-practical for many systems with unknown number of participating nodes in advance.

Our proposed coding solution differs from the regenerative codes [26] [25] [27] [28] [29] as well. The structures of these codes also depend on the number of participating nodes for recovery. The focus of regenerative codes are on a tradeoff between the storage cost (amount of data is being stored in each node) and the repair bandwidth. In one side of this tradeoff, the *Minimum Storage Regenerating (MSR)* codes seek the optimal repair bandwidth for minimum storage cost (storing M/k amount of data in each node). [29] and [30] considered the problem of I/O overhead in addition to the repair bandwidth metric and proposed a solution that significantly reduces the

I/O access. While the I/O access cost is not the focus of our paper, the proposed designs in these papers fail to work when the number of available nodes, *helpers*, changes.

We introduce a new class of MDS codes using a search algorithm that we call rotation codes with progressive engagement property. Moreover, we illustrate how the original recovery (decoding) scheme of the existing permutation code featured in [21] can be modified to accommodate the progressive engagement property.

The rest of the paper is organized as follows. Section II describes the system model and state the problem of existing MDS codes with optimal repair bandwidth. The concept of progressive engagement for the MDS codes is introduced in Section III along with its rigorous definition. Moreover, two examples of MDS codes with progressive engagement property are introduced in this section. Numerical results and conclusions are provided in Sections IV and V, respectively.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a **symmetric** storage system consisting of an array of total n disks or nodes¹, among which k systematic nodes and $n - k = m$ parity nodes. Systematic nodes hold the original data. Suppose a file of size M is evenly partitioned into k parts, each of size $L = \frac{M}{k}$ blocks, and stored in the k systematic nodes. Each block also consists of a fixed number of w -bit symbols. The content of parity nodes is calculated from the original data using an erasure encoder. The coding and decoding operations are conducted over the Galois Field 2^w , where $n < 2^w - 1$. The common values for w are 8, 16, 32 depending on the size of the network. We assume an erasure code (n, k) with MDS property meaning that any k nodes suffice to recover the original data. This also means that up to $(n - k)$ node failures can be tolerated without data loss. An example of such system with $k = 2$ systematic nodes S-node 1 and S-node 2, $m = 2$ parity nodes P-node 1 and P-node 2, and $L = 2$ blocks is depicted in Fig. 1.

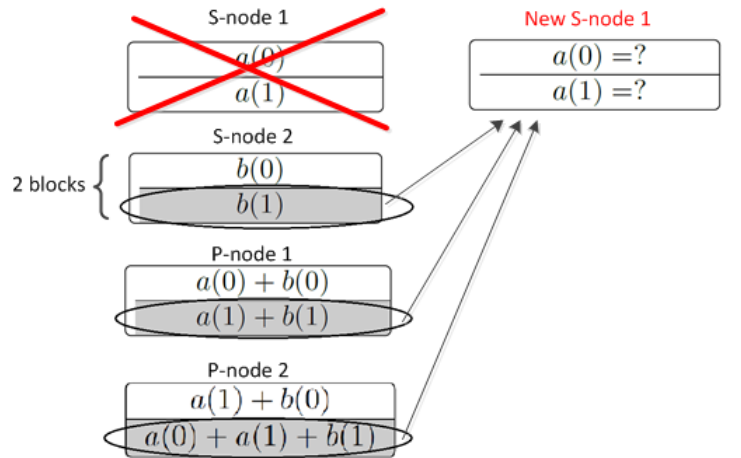


Fig. 1: Exact repair for $n = 4$, $k = 2$, and $L = 2$ blocks ($M = 4$ blocks)

¹We use node and disk interchangeably throughout this paper.

To maintain the data integrity, when a node fails, the lost data must be recovered and saved into a new node. For better exposition of this issue, assume the first node fails in a system with $n = 4$, $k = 2$, $L = 2$ [31]. Using a traditional MDS code, the new node could contact any 2 nodes and download 4 blocks of data, from which, $a(0)$ and $a(1)$ can be obtained. However, with the encoding scheme shown in Fig. 1, it is also possible to do the following decoding to recover the lost data in Node 1: we can download only three blocks $b(1)$; $a(1) + b(1)$; $a(0) + a(1) + b(1)$, from the three surviving nodes, and attain $a(0)$ and $a(1)$ via solving a linear equation. In this case, the repair bandwidth, defined as the minimum amount of data that must be downloaded to recover the lost data, is 3 blocks as opposed to 4 for the traditional MDS code.

Several work exist that propose MDS codes with minimum repair bandwidth [25], [20], [21], [32], [26], [25], [27]. Let d denote the number of surviving nodes participating in the recovery process. For these codes to work, d is a parameter that must be known when designing a code. Moreover, most of these designs only exist for a particular d with $d = n - 1$ being the most common case. In this case, if d decrements by one (for example, a node becomes unavailable), in order to obtain the repair-bandwidth gain, a new code with $(n', k', d') = (n - 1, k, d - 1)$ must be loaded to the storage nodes, as if the unavailable node had not existed in the first place. The structures of these two codes can be quite different. To shed a light on this problem, consider the permutation code in [21] for the case $(n, k, d) = (5, 2, 4)$ in Table I. If d is set to 3 in this code, the specific recovery scheme in [21] no longer works and we have to change the whole content of the parity nodes to that of the $(n', k', d') = (4, 2, 3)$ in Table II to be able to recover the lost data. As can be seen, these two codes are completely different. Such a recalculation and reloading of the content cannot be tolerated in many practical cases. The parameters λ_i 's in Tables I and II are chosen so that the MDS properties are satisfied in these codes.

In this paper we consider a single failure scenario and assume that all the $k - 1$ remaining systematic nodes participate for recovering the lost data. We denote the repair bandwidth by $\gamma(p)$ where $p = d - (k - 1)$ for a given coding scheme. It is noted that most of our definitions and proposed schemas are applicable to the general case where any d nodes are selected for recovery.

In the next section, we will introduce the concept of 'progressive engagement' to address the problem stated above.

III. PROGRESSIVE ENGAGEMENT PROPERTY

We aim to design a family of MDS codes (n, k, d) for $d = k, \dots, n - 1$ with the same encoding that while achieving low repair bandwidth it can be used when d (or equivalently p) is not priori known. We call this property of MDS codes *progressive engagement*. An MDS code with progressive engagement property provides flexibility in choosing the optimal set of participating nodes without having to change the coding scheme and the content of the storage nodes. We provide the following formal definition for such codes.

Definition 1: A family of MDS codes (n, k, d) , where $d = k, \dots, n - 1$, with L blocks provides the progressive engagement property if

- 1) The encoder takes kL blocks, from the k systematic nodes, and provides $(n - k)L$ parity blocks for the $(n - k)$ parity nodes.
- 2) The decoder **can** access dL blocks from $d = k, \dots, n - 1$ surviving nodes and generate the L missing blocks.
- 3) Let us define $p = d - k + 1$, where $p = 1, \dots, n - k$. The decoder will only **use** $\gamma(p) \leq dL$ blocks of data to generate the L missing blocks.
- 4) For $p < p'$, we have $\bar{\gamma}(p) < \bar{\gamma}(p')$ where $\bar{\gamma}$ is the average of repair bandwidth taken over all p participating parity nodes for the recovery.

For any value of d (correspondingly p with the assumption that all systematic surviving nodes participate for recovery), we hope that such a code would require as minimum repair bandwidth as if it were designed for that particular d . However, even if that minimum bandwidth is not achieved for all d , the MDS codes with progressive engagement property are still practically more amenable over the codes that only work for a fixed d .

While the progressive engagement property is practically appealing from many aspects as mentioned earlier, it is also motivated from the following proposition.

Proposition 1: Once a failure occurs, the best strategy to minimize a weighted sum of the *accessing cost* and the repair bandwidth to recover the lost data is to incrementally engage the parity nodes according to their accessing costs until both repair bandwidth and total accessing cost constraints are met.

Proof: The proof is provided in Appendix A ■

The advantage of the MDS codes with progressive engagement property over the traditional MDS codes such as Reed-Solomon codes is the capability of reducing repair bandwidth by involving more participating nodes. In fact, traditional MDS codes fail to satisfy the second condition of Definition 1, while the recently proposed codes that yield minimum repair bandwidth fail to meet the first condition. For the codes [25], [20], [21], [32] the number of participating nodes must be known when designing the codes. As d may not be known a priori, it is a great advantage of the MDS codes with progressive engagement property that the codes can be used when d varies without changing the code structure or the content of the nodes.

An analogy exists between the codes with progressive engagement property and rate-compatible punctured error correcting codes [24]. The rate-compatible punctuated codes are very important in wireless communication since they allow transmitting information with various rates without redesigning the code while the code is as efficient as if it were designed for the respective rate. Similarly, the progressive engagement property ensures to recover the lost data using any number of surviving nodes without redesigning the codes.

In what follows, we will provide two examples of MDS codes with progressive engagement property.

TABLE I: Permutation code construction for $(n, k, d) = (5, 2, 4)$. The gray and green blocks are used in Section III-B to describe the recovery scheme.

| S-node 1 | S-node 2 | P-node 1 | P-node 2 | P-node 3 |
|----------|----------|-----------------------------------|---------------------------------------|---------------------------------------|
| $a(0)$ | $b(0)$ | $\lambda_1 a(0) + \lambda_2 b(0)$ | $\lambda_1^2 a(3) + \lambda_2^2 b(1)$ | $\lambda_1^3 a(6) + \lambda_2^3 b(2)$ |
| $a(1)$ | $b(1)$ | $\lambda_1 a(1) + \lambda_2 b(1)$ | $\lambda_1^2 a(4) + \lambda_2^2 b(2)$ | $\lambda_1^3 a(7) + \lambda_2^3 b(0)$ |
| $a(2)$ | $b(2)$ | $\lambda_1 a(2) + \lambda_2 b(2)$ | $\lambda_1^2 a(5) + \lambda_2^2 b(0)$ | $\lambda_1^3 a(8) + \lambda_2^3 b(1)$ |
| $a(3)$ | $b(3)$ | $\lambda_1 a(3) + \lambda_2 b(3)$ | $\lambda_1^2 a(6) + \lambda_2^2 b(4)$ | $\lambda_1^3 a(0) + \lambda_2^3 b(5)$ |
| $a(4)$ | $b(4)$ | $\lambda_1 a(4) + \lambda_2 b(4)$ | $\lambda_1^2 a(7) + \lambda_2^2 b(5)$ | $\lambda_1^3 a(1) + \lambda_2^3 b(3)$ |
| $a(5)$ | $b(5)$ | $\lambda_1 a(5) + \lambda_2 b(5)$ | $\lambda_1^2 a(8) + \lambda_2^2 b(3)$ | $\lambda_1^3 a(2) + \lambda_2^3 b(4)$ |
| $a(6)$ | $b(6)$ | $\lambda_1 a(6) + \lambda_2 b(6)$ | $\lambda_1^2 a(0) + \lambda_2^2 b(7)$ | $\lambda_1^3 a(3) + \lambda_2^3 b(8)$ |
| $a(7)$ | $b(7)$ | $\lambda_1 a(7) + \lambda_2 b(7)$ | $\lambda_1^2 a(1) + \lambda_2^2 b(8)$ | $\lambda_1^3 a(4) + \lambda_2^3 b(6)$ |
| $a(8)$ | $b(8)$ | $\lambda_1 a(8) + \lambda_2 b(8)$ | $\lambda_1^2 a(2) + \lambda_2^2 b(6)$ | $\lambda_1^3 a(5) + \lambda_2^3 b(7)$ |

TABLE II: Permutation code construction for $(n, k, d) = (4, 2, 3)$

| S-node 1 | S-node 2 | P-node 1 | P-node 2 |
|----------|----------|-----------------------------------|---------------------------------------|
| $a(0)$ | $b(0)$ | $\lambda_1 a(0) + \lambda_2 b(0)$ | $\lambda_1^2 a(2) + \lambda_2^2 b(1)$ |
| $a(1)$ | $b(1)$ | $\lambda_1 a(1) + \lambda_2 b(1)$ | $\lambda_1^2 a(3) + \lambda_2^2 b(0)$ |
| $a(2)$ | $b(2)$ | $\lambda_1 a(2) + \lambda_2 b(2)$ | $\lambda_1^2 a(0) + \lambda_2^2 b(3)$ |
| $a(3)$ | $b(3)$ | $\lambda_1 a(3) + \lambda_2 b(3)$ | $\lambda_1^2 a(1) + \lambda_2^2 b(2)$ |

A. Progressive Engagement in Rotation Codes

In this section, we propose a class of MDS codes that satisfies the progressive engagement property and design the corresponding encoding and decoding schemes. We propose a computer search to find a code with the lowest repair bandwidth when different parity nodes participate in a single failure recovery. To make the computer search feasible, we reduce the search space by constraining the encoder to be a member of a family of codes that we call *rotation codes*. A family of (n, k, d) , $d = k, \dots, n-1$, rotation codes with length L is defined as follows. Since our proposed code is independent of d , we may skip d in the code representation parameters. Let us define \mathbf{a}_i , $i = 1, \dots, k$ as the k systematic node vectors. Then, the $n-k$ parity node vectors are defined as

$$\mathbf{p}_j = \sum_{i=1}^k \lambda_{ij} \mathbf{a}_i \mathbf{R}_{ij}, \quad j = 1, \dots, n-k \quad (1)$$

where the coefficients λ_{ij} are chosen to ensure the MDS property and \mathbf{R}_{ij} is a rotation matrix defined below. Consider $\mathbf{I}_L = [\mathbf{e}_1; \mathbf{e}_2; \dots; \mathbf{e}_L]$ as the $L \times L$ identity matrix, where \mathbf{e}_l is a row vector of length L with 1 in the position l and 0 in every other positions. Also, we define $\mathbf{R}_1 = [\mathbf{e}_{1+1}; \mathbf{e}_{1+2}; \dots; \mathbf{e}_L; \dots; \mathbf{e}_1]$ by cyclically rotating \mathbf{I}_L 's rows l times. Then, $\mathbf{R}_{ij} = \mathbf{R}_1$ for some l .

To search for a good code, one can try all possible \mathbf{R}_{ij} . Note that there are $k(n-k)$ positions for \mathbf{R}_{ij} and the maximum complexity for a full search is $L^{k(n-k)}$ and finite. However, usually $\mathbf{R}_{i1} = \mathbf{R}_{1j} = \mathbf{I}_L$, $i = 1, \dots, k$, $j = 1, \dots, n-k$ and one can use other symmetrical properties to further reduce the search space. For a given set of rotation matrices \mathbf{R}_{ij} , we choose λ_{ij} parameters to make the code an MDS code. This can be done by writing the MDS constraint equations and solving them for one set of λ_{ij} parameters. The solution is not unique, but any choice of λ_{ij} parameters as long as the code is an MDS code is acceptable. The next step is to check if the code provides the progressive engagement property. To

check the property, we consider all possible single failure cases and all possible involved parity nodes and calculate the corresponding repair bandwidth. If the repair bandwidth values are a decreasing function of the number of surviving nodes, i.e. involved parity nodes, the code is acceptable. In other words, for each code, we need to prove that in the case of single failure, the lost data can be re-constructed using any number of parity nodes. Since the above rotation code is an XOR-based code, we can employ the algorithm proposed in [33] for single failure recovery while achieving low repair bandwidth. This algorithm aims to minimize the repair bandwidth regardless of λ_i 's as long as they are non-zero. More specifically, for any systematic node failure and any participating set of surviving nodes, this algorithm constructs a directed weighted graph in which the shortest path between the root and any leaf (all leaves are connected) would determine the data blocks that must be downloaded and the minimum repair bandwidth. The height of this tree is equal to the number of blocks in the failed node. Using this algorithm, one can verify if both conditions of the progressive engagement property are satisfied for each choice of the code in (1). Note that the repair bandwidth resulted from this algorithm may not be the minimum but it is usually close to the minimum. To describe the details, we concentrate on an example for $(n, k) = (6, 3)$ rotation code with $L = 4$ as a candidate:

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{a} + \mathbf{b} + \mathbf{c} \\ \mathbf{p}_2 &= \mathbf{a} + \lambda_1 \mathbf{R}_1 \mathbf{b} + \lambda_2 \mathbf{R}_3 \mathbf{c} \\ \mathbf{p}_3 &= \mathbf{a} + \lambda_1^2 \mathbf{R}_2 \mathbf{b} + \lambda_2^2 \mathbf{R}_1 \mathbf{c} \end{aligned} \quad (2)$$

where $\mathbf{R}_1 = [\mathbf{e}_2; \mathbf{e}_3; \mathbf{e}_4; \mathbf{e}_1]$, $\mathbf{R}_2 = [\mathbf{e}_3; \mathbf{e}_4; \mathbf{e}_1; \mathbf{e}_2]$ and $\mathbf{R}_3 = [\mathbf{e}_4; \mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3]$ are the rotation matrices with \mathbf{e}_i being a row vector of length 4 with 1 in the position i and 0 in every other positions. The coefficients λ_1 and λ_2 in (2) are chosen to ensure the MDS property. For $\lambda_1 = 2$ and $\lambda_2 = 3$ this property is explicitly proven in Appendix B. The resulting structure is depicted in Table III. Next, we show that the code in Table III provides the progressive engagement property. As explained

before, we use the algorithm proposed in [33] to recover the single failure with minimum repair bandwidth. The result of this algorithm is presented in Table IV when S-node 1 fails and for all the sets of participating parity nodes.

As can be seen in Table IV, there are two sets with the same number of participating nodes that yield different repair bandwidths. Taking average over all failure and participating set cases, we have

$$\bar{\gamma}(p=1) = 12, \quad \bar{\gamma}(p=2) = 8.66, \quad \bar{\gamma}(p=3) = 8$$

It is evident that the rotation code in (2) provides the progressive engagement property as the number of participating parity nodes increases, the repair bandwidth drops.

B. Progressive Engagement in Permutation Code

In this section, we consider the code structure based on the permutational matrix featured in [21]. This code with its original recovery scheme (decoding) cannot meet the progressive engagement property in Definition 1. However, we will describe how the recovery scheme can be modified to allow the flexible engagement of parity nodes.

Proposition 2: The permutation code featured in [21] with the modified recovery scheme described below provides the progressive engagement property. Furthermore, the repair bandwidth cost for a given p is

$$\gamma(p) = kL - \frac{L}{n-k}(p-1)(k-1), \quad (3)$$

where $L = (n-k)^k$.

Proof:

We begin for the case with $n = 5$, $k = 2$ and $L = 9$ to illustrate the basic idea of decoding that results in the progressive engagement property. Following the permutation code structure in [21], we reach to the code depicted in Table I for 2 systematic nodes and 3 parity nodes. λ_1 and λ_2 are two non-zero scalars in the field $\mathcal{F}(2^w)$ with $2^w \geq 2k+1 = 5$ so that

$$\lambda_1 + \lambda_2 \neq 0, \quad \lambda_1 \neq \lambda_2. \quad (4)$$

For simplicity, consider the case where S-node 1 fails. The original recovery scheme only considers the case where all surviving nodes are engaged to recover the single failure. In this case, the new node just needs to download the first $\frac{1}{n-k} = \frac{1}{3}$ fraction of the total block rows of every surviving node. This fraction is indicated by the shaded gray region in Table I, i.e., S-nodes 2 and P-nodes 1, 2, 3. The downloaded blocks result in 12 equations and 12 unknowns, among which 9 lost data $a(0), \dots, a(8)$ can be readily obtained. Hence, the total repair-bandwidth

$$\gamma(p=3) = (n-1) \times \frac{1}{n-k}L = 12$$

suffices for recovering the lost data. A similar recovery can be done when S-node 2 fails with the only difference that the block rows l with $(l \bmod 3 = 1)$ must be downloaded from every surviving node.

Next, we consider the case where only two parity nodes P-node 1 and P-node 2 beside S-node 2 are selected. In this

case, the new node still downloads the first three blocks of the available nodes S-node 2 and P-nodes 1 and 2. From these 9 equations and 9 unknowns, we can reconstruct the blocks of vector a appeared in the downloaded blocks, i.e., $a(0), a(1), a(2)$ and $a(3), a(4), a(5)$. The remaining blocks $a(6), a(7), a(8)$ can be obtained by downloading the next three blocks of P-node 2 and S-node 1 (green region in Table I). This strategy results in $\gamma(p=2) = 3 \times \frac{L}{n-k} + 2 \times \frac{L}{n-k} = 15$ total repair bandwidth for $p=2$ or $d = p + (k-1) = 3$. We note that while the achieved repair bandwidth may not be the minimum repair bandwidth possible, it is lower than the total file size $M = 18$ blocks. We note that a similar strategy would also work if the parity nodes other than P-node 1 and P-node 2 were selected. For instance, for P-node 2 and P-node 3, in addition to the first 3 rows of these nodes and S-node 2, by downloading the last three blocks of S-node 2 and P-node 2, we can reconstruct the whole $a(0), \dots, a(8)$. For $p=1$ we again download the first three blocks of that parity node along with the first three blocks of S-node 2, thus obtaining the respected components of a saved on those blocks. For the remaining 6 blocks of a , we can simply download the remaining 6 blocks of S-node 2 and the remaining 6 blocks of the selected parity node, which in total results in $\gamma(p=1) = 2 \times \frac{L}{n-k} + 2 \times 3 \times \frac{L}{n-k} = 18$.

A similar argument can be made for the general case of the permutation codes. The proof of (3) for any permutation code with the modified recovery scheme is provided in Appendix C. ■

It remains unknown whether $\gamma(p)$ in (3) is the optimal repair bandwidth for the codes that support progressive engagement. The lower bound for the exact repair with $d = p + k - 1$ participants is obtained in [19]

$$\gamma_{\min}(p) = \frac{L(p+k-1)}{p}. \quad (5)$$

In the simulation section, we provide a comparison between this lower bound and the repair bandwidth of our proposed codes.

IV. SIMULATION RESULTS

In this section, we provide some numerical experiments to assess the performance of the rotation and permutation codes.

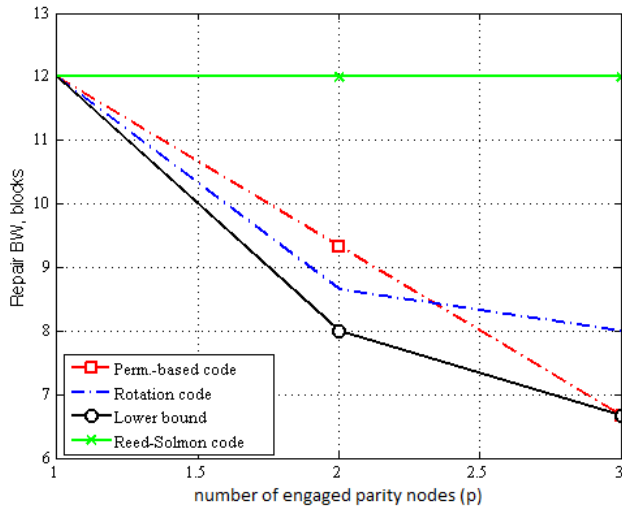
As the first experiment, it would be insightful to compare the lower-bound (5) and the repair bandwidth of the rotation code and permutation code with our modified recovery scheme in (3). Fig. 2 provides such a repair bandwidth comparison and depicts the number of blocks versus the number of participating parity nodes p for $(n, k) = (6, 3)$. For completeness of comparison, the repair bandwidth required by Reed-Solomon code is also included in Fig. 2. Our proposed rotation code and the permutation code with our modified recovery scheme significantly outperform the Reed-Solomon code in terms of repair bandwidth saving. Furthermore, both of these codes satisfy the progressive engagement property as their associated repair bandwidth drops as p grows and do not need a new code structure and/or new node contents when p changes. We also observe that our proposed rotation code

TABLE III: Rotation code construction for $(n, k) = (6, 3)$ and $L = 4$ with progressive engagement property

| S-node 1 | S-node 2 | S-node 3 | P-node 1 | P-node 2 | P-node 3 |
|----------|----------|----------|----------------------|------------------------|----------------------------|
| $a(0)$ | $b(0)$ | $c(0)$ | $a(0) + b(0) + c(0)$ | $a(0) + 2b(1) + 3c(3)$ | $a(0) + 2^2b(2) + 3^2c(1)$ |
| $a(1)$ | $b(1)$ | $c(1)$ | $a(1) + b(1) + c(1)$ | $a(1) + 2b(2) + 3c(0)$ | $a(1) + 2^2b(3) + 3^2c(2)$ |
| $a(2)$ | $b(2)$ | $c(2)$ | $a(2) + b(2) + c(2)$ | $a(2) + 2b(3) + 3c(1)$ | $a(2) + 2^2b(0) + 3^2c(3)$ |
| $a(3)$ | $b(3)$ | $c(3)$ | $a(3) + b(3) + c(3)$ | $a(3) + 2b(0) + 3c(2)$ | $a(3) + 2^2b(1) + 3^2c(0)$ |

 TABLE IV: Decoding strategy for a $(n, k) = (6, 3)$ code with $L = 4$ when node S-1 fails.

| Accessible parity set | Optimal parity blocks downloaded | Repair bandwidth |
|-------------------------------|--|----------------------|
| P-node1 or P-node2 or P-node3 | all blocks in the selected parity node | $\gamma(p = 1) = 12$ |
| (P-node1, P-node2) | $p_2(3), p_2(1), p_1(2), p_1(0), c(0), c(2), b(0), b(2)$ | $\gamma(p = 2) = 8$ |
| (P-node1, P-node3) | $p_3(3), p_3(2), p_1(0), p_1(1), c(0), c(1), c(3), b(0), b(1)$ | $\gamma(p = 2) = 9$ |
| (P-node2, P-node3) | $p_2(3), p_2(2), p_2(1), p_2(0), c(0), c(3), b(0), b(1), b(2)$ | $\gamma(p = 2) = 9$ |
| (P-node1, P-node2, P-node3) | $p_2(3), p_2(1), p_1(2), p_1(0), c(0), c(2), b(0), b(2)$ | $\gamma(p = 3) = 8$ |


 Fig. 2: Repair bandwidth versus the number of parity nodes participating in recovery for the rotation code in (1) and permutation code with modified recovery scheme and Reed-Solomon code $(n, k) = (4, 2)$.

results in lower repair bandwidth compared to the permutation code with modified decoding when only two parity nodes are available for recovery. However, when all three parity nodes are involved, the permutation code requires the same bandwidth as the lower bound and lower than that of our proposed rotation code.

In the next experiment, we consider the $(n, k) = (10, 3)$ permutation code and want to evaluate the speed of recovering the data of a failed node versus the number of participating nodes for three symbol sizes $w = 8, 16, 32$.² We implement this MDS code in C using the open source library for Galois Field arithmetic in [35]. We assume an $M = 32$ MB file is distributed across $n = 10$ storage nodes with $m = 7$ parity nodes. This follows $L = m^3 = 343$ blocks of size roughly $\frac{M}{Lk} = 32KB$. The speed here is defined as the amount of data recovered per second. On our 1.8 GHz Intel core i5, we obtain the results shown in Fig. 3 using Monte Carlo simulation with 100 iterations. As observed from this figure, the recovery speed increases as more parity nodes are engaged. Moreover,

although arithmetic calculation in $GF(2^{32})$ is more time-consuming than $GF(2^8)$ and $GF(2^{16})$, the case corresponding to $w = 32$ demonstrates the highest speed among the three cases. This is due to the fact that for a fixed block size, the case of $w = 32$ bits has a lot less symbols to process compared to the other cases.

In the previous experiment, the accessing cost of parity nodes was ignored in favor of just showing the recovery speed of the code. In the next experiment, following the definition and terminologies of accessing costs in Appendix A, we include the accessing cost in terms of the number of hops for a system whose 7 parity nodes are located from 1 to 7 hops away, *i.e.*, $c_{1i} = i$. For such a system, Fig. 4 plots the total cost versus the number of participating nodes for various coding schemes using two weights. The proposed solution in Section II readily obtains the optimal p^* in our code with progressive engagement property (denoted by Prog. Engag). This optimal point clearly outperforms the Reed-Solomon code and the codes designed for minimizing the repair bandwidth in terms of achieving a lower total cost. Moreover, unlike our code and Reed-Solomon code, the codes designed for minimizing repair bandwidth are just two points in this plane instead of a curve and have no values for p other than $p = (n - k) = 7$.

V. CONCLUSIONS

We introduced a new class of MDS codes with a property called progressive engagement. This property provides flexibility in engaging more surviving nodes in favor of reducing the repair bandwidth without redesigning the code structure and content of the existing nodes. We argued that such property is not met in the existing MDS codes. We further introduced a new class of MDS codes, called rotation codes, with progressive engagement property. Moreover, we illustrated how the existing permutation codes can provide progressive engagement by modifying the original recovery scheme. The repair bandwidth of the rotation codes and permutation codes were calculated and compared with other MDS codes.

²This case is the default rate for cloud file system Tahoe-LAFS [34].

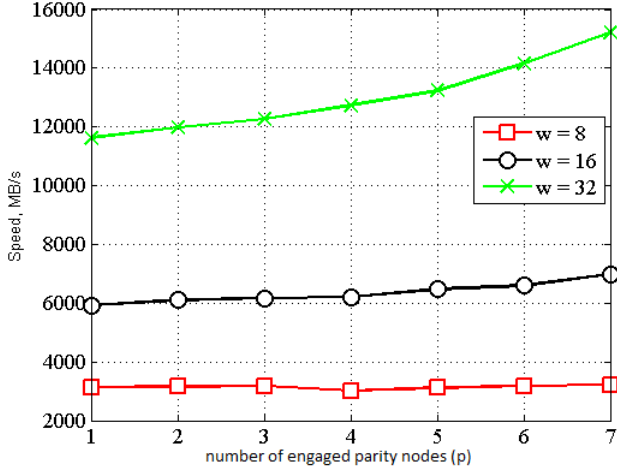


Fig. 3: CPU speed of single failure recovery versus the number of participating nodes for permutation code with $(n, k) = (10, 3)$

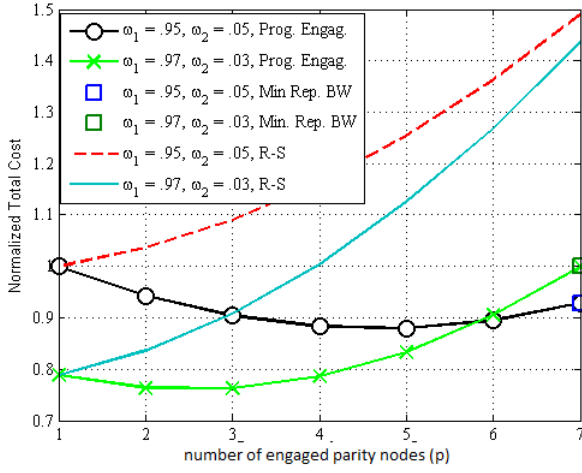


Fig. 4: Normalized total cost versus the number of participating nodes where $(n, k) = (10, 3)$ for various weights

APPENDIX A FAILURE RECOVERY IN THE PRESENCE OF ACCESSING COST

In this appendix, we provide the best strategy to recover the lost data of the failed node when both accessing cost and repair bandwidth matter.

It is not desirable to engage all the nodes in a cloud system with a large number of geographically distributed nodes to recover the lost data. It is of our interest to engage an optimal set of available nodes for the recovery process considering their distances, link throughput as well as the total repair bandwidth. Subsequently, we consider two specific cost metrics for an optimal selection strategy; the accessing cost and the repair bandwidth cost.

The accessing cost is represented by the cost matrix $\mathbf{C} = [c_{ij}]_{n \times n}$, where c_{ij} denotes the cost associated with contacting and accessing Node j for recovering the data of Node i . The value of c_{ij} can be a function of multiple factors including communication delay, the number of hops to reach Node j

and so on. The sum cost of all nodes of a set determines the total accessing cost of that set.

Our objective is then to find p^* nodes from the parity set P-nodes $1, \dots, m$ that minimize the weighted sum of the repair bandwidth and the accessing cost. Note that the case where the objective is to minimize one of these cost metrics subject to a constraint on the other metric can be converted to a weighted sum metric with Lagrange multipliers as the weights. To proceed, we define a binary indicator $\alpha_i \in \{0, 1\}$, $\forall i = 1, \dots, m$ such that α_i is 1 if P-node i is selected, and 0 otherwise. Without loss of generality, assume that the systematic node S-node 1 fails. Then, this problem can be cast into the following integer programming problem

$$\min_{\alpha_1, \dots, \alpha_m} \omega_1 \sum_{i=1}^m \alpha_i c_{1i} + \omega_2 \gamma \left(\sum_{i=1}^m \alpha_i \right) \quad (6)$$

$$s.t. \quad I) \quad \sum_{i=1}^m \alpha_i \geq 1 \quad (7)$$

$$II) \quad \alpha_i \in \{0, 1\}, \quad i = 1, \dots, m,$$

where the first and second terms in (6) encompass the accessing cost and the repair bandwidth cost, respectively, ω_1 and ω_2 are the respective weights normalized to $\omega_1 + \omega_2 = 1$. Condition I in (6) requires at least one parity node to be selected in order to ensure the reconstruction of the lost data for any MDS code.

At the first glance, Problem (6) may seem to be hard to solve. However, the solution becomes apparent by letting $\sum_{i=1}^m \alpha_i = p$. Given p , the optimization variables can be obtained from

$$A(p) = \min_{\alpha_1, \dots, \alpha_m} w_1 \sum_{i=1}^m \alpha_i c_{1i} \quad (8)$$

$$s.t. \quad I) \quad \sum_{i=1}^m \alpha_i = p \quad (9)$$

$$II) \quad \alpha_i \in \{0, 1\}, \quad i = 1, \dots, m$$

with solution

$$\alpha_i(p)^* = \begin{cases} 1, & \text{if } i \in \arg \min_p (c_{11}, \dots, c_{1m}) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $\arg \min_p (c_{11}, \dots, c_{1m})$ returns an index set of the first p smallest c_{11}, \dots, c_{1m} and can be easily done by sorting the accessing costs. Note that, so far, without any loss of generality, we have assumed the systematic node S-node 1 has failed. In order to avoid the above computation for each p , we can just sort the accessing costs c_{11}, \dots, c_{1m} for any failed node l once with complexity $m \log m$ using the quicksort algorithm and just return the index of the first p elements. Note that the $w_2 \gamma(p)$ was dropped from (6) since, given p , it was a constant term. As p can take only integer values from 1 to m , the solution to (6) can be obtained by iterating over $p = 1, \dots, m$ and finding p^* that achieves minimum $A(p) + w_2 \gamma(p)$. Once p^* is known, $\alpha_i(p^*)^*$ determines whether parity node i is in the optimal selection set or not. The overall search complexity in this case is in the order of $O(m + m \log m)$.

The optimization procedure above leads us to a selection strategy that incrementally engages the parity nodes according to their order of accessing cost. After each new engagement, it calculates the total cost of accessing and repair bandwidth.

APPENDIX B

MDS PROPERTY OF THE ROTATION CODE IN (2)

We need to prove that the coefficient matrix associated with any 3 nodes selected from the total 6 nodes is full rank. We consider the following cases:

Case 1: 3 Systematic Nodes Selected

It is obvious that the coefficient matrix associated with this selection is full rank.

Case 2: 2 Systemic Nodes and 1 Parity Node Selected

This case is also trivial. For instance, consider S-node 1, S-node 2 and P-node 3 are selected. The rank of the coefficient matrix is

$$\text{rank} \left(\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & 2^2\mathbf{R}_2 & 3^2\mathbf{R}_3 \end{bmatrix} \right) = Lk = 12 \quad (11)$$

which is full rank.

Case 3: 1 Systemic Node and 2 Parity Nodes selected

First, let us verify the full rank coefficient matrix for a case where S-node 1, P-node 2 and P-node 3 are selected. We need to show that the determinant of the associated coefficient matrix is non-zero. We have

$$\begin{vmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & 2\mathbf{R}_1 & 3\mathbf{R}_3 \\ \mathbf{I} & 2^2\mathbf{R}_2 & 3^2\mathbf{R}_1 \end{vmatrix} = \begin{vmatrix} 2\mathbf{R}_1 & 3\mathbf{R}_3 \\ 2^2\mathbf{R}_2 & 3^2\mathbf{R}_1 \end{vmatrix} = \quad (12)$$

$$|2\mathbf{R}_1| |3^2\mathbf{R}_1 - 2^2\mathbf{R}_2(2\mathbf{R}_1)^{-1}3\mathbf{R}_3| = 6|3\mathbf{R}_1 - 2\mathbf{I}| \neq 0 \quad (13)$$

where from (12) to (13) we have used the following equation from the matrix theory

$$\begin{vmatrix} \mathbf{P} & \mathbf{S} \\ \mathbf{R} & \mathbf{Q} \end{vmatrix} = |\mathbf{P}| \cdot |\mathbf{Q} - \mathbf{R}\mathbf{P}^{-1}\mathbf{S}|. \quad (14)$$

Furthermore, equations $\mathbf{R}_1^{-1} = \mathbf{R}_3$, $\mathbf{R}_2\mathbf{R}_3 = \mathbf{R}_1$ and $\mathbf{R}_1\mathbf{R}_3 = \mathbf{I}$ were used to derive (13). A similar derivation is applicable to the other selection cases of 1 systematic node and 2 parity nodes.

Case 4: 3 Parity Nodes Selected

In this case, the determinant of the coefficient matrix will be

$$\begin{vmatrix} \mathbf{I} & \mathbf{I} & \mathbf{I} \\ \mathbf{I} & 2\mathbf{R}_1 & 3\mathbf{R}_3 \\ \mathbf{I} & 2^2\mathbf{R}_2 & 3^2\mathbf{R}_1 \end{vmatrix} = \begin{vmatrix} 2\mathbf{R}_1 - \mathbf{I} & 3\mathbf{R}_3 - \mathbf{I} \\ 2^2\mathbf{R}_2 - \mathbf{I} & 3^2\mathbf{R}_1 - \mathbf{I} \end{vmatrix} = |2\mathbf{R}_1 - \mathbf{I}|$$

$$|3^2\mathbf{R}_1 - \mathbf{I} - (2^2\mathbf{R}_2 - \mathbf{I})(2\mathbf{R}_1 - \mathbf{I})^{-1}(3\mathbf{R}_3 - \mathbf{I})| \neq 0 \quad (15)$$

which can be easily verified to be non-zero.

APPENDIX C

PROOF OF PROPOSITION 2 FOR GENERAL CASE (n, k)

We describe how the recovery scheme can be generalized to the case with arbitrary (n, k) and variable p . Without loss of generality, suppose P-node 1 fails. The recovery scheme involves two phases. In the first phase, we download the same block rows of the given p selected parity nodes and the $k - 1$ systematic nodes as the original recovery scheme would do with the assumption of full participation. In total, we get $(k - 1) \times \frac{L}{n-k} + p \times \frac{L}{n-k}$ equations with the same number of unknowns, among which, $p \times \frac{L}{n-k}$ components belong to Node 1 and can be recovered. The second phase is meant to recover the rest of the $((n - k) - p) \frac{L}{n-k}$ components of the parity Node 1. For that, we consider any arbitrary parity node (e.g. P-node j) from the p given parity nodes and all $k - 1$ surviving systematic nodes. We then download $((n - k) - p) \frac{L}{n-k}$ block rows from P-node j containing the components of P-node 1 that were missing in the first recovery phase. Next, we download all the components of these downloaded vectors from their respective systematic nodes. This would give us enough number of equations to recover the missing components of P-nodes in the first phase. Hence the total repair bandwidth for a given p is

$$\gamma(p) = (p + (k - 1)) \frac{L}{n - k} + k \left(((n - k) - p) \right) \frac{L}{n - k} =$$

$$\frac{-(p - 1)(k - 1) + k(n - k)}{n - k} L = kL - \frac{L}{n - k} (p - 1)(k - 1).$$

REFERENCES

- [1] L. Donald, *MCSA/MCSE 2003 JumpStart: Computer and Network Basics*. Sybex, 2003.
- [2] V. Pless, F. MacWilliams, N. Sloane, R. Blahut, and R. McEliece, *Introduction To The Theory Of Error-correcting Codes*, 3rd, july 1998.
- [3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, *et al.*, "Erasure coding in windows azure storage," in *USENIX ATC*, 2012.
- [4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, 2003.
- [5] A. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.
- [6] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. of the Society for Industrial & Applied Math.*, vol. 8, no. 2, pp. 300–304, June 1960.
- [7] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar, "Row-diagonal parity for double disk failure correction," in *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST)*, 2004.
- [8] M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in RAID architectures," *IEEE Trans. on Computer*, vol. 44, no. 2, pp. 192–202, Feb. 1995.
- [9] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 272–276, 1999.
- [10] J. L. Hafner, "WEAVER codes: Highly fault tolerant erasure codes for storage systems," in *Proc. of the 4th USENIX on Conference on File and Storage Tech. (FAST)*, vol. 5. Berkeley, CA, USA: USENIX Association, March 2005.
- [11] M. Luby, "LT codes," Digital Fountain, Inc. luby@digitalfountain.com, 2002.
- [12] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.
- [13] J. S. Plank and M. G. Thomason, "On the practical use of LDPC erasure codes for distributed storage applications," *University of Tennessee, Tech. Rep. CS-03-510*, 2003.

- [14] J. J. Wylie and R. Swaminathan, "Determining fault tolerance of XOR-based erasure codes efficiently," in *37th Annual IEEE/IFIP International Conf. on Dependable Systems and Networks*, June 2007.
- [15] J. J. Wylie, "Finding the most fault-tolerant flat XOR-based erasure codes for storage systems," in *45th Signals, Systems and Computers (ASILOMAR) Conference*. IEEE, 2011.
- [16] F. Oggier and A. Datta, "Self-repairing codes for distributed storage projective geometric construction," in *Information Theory Workshop (ITW)*, IEEE, 2011.
- [17] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *Information Theory, IEEE Transactions on*, vol. 60, no. 10, pp. 5843–5855, 2014.
- [18] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*, ser. ISIT'09, 2009, pp. 2276–2280.
- [19] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.
- [20] N. Shah, K. Rashmi, P. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Information Theory Workshop (ITW)*, 2010.
- [21] V. R. Cadambe, C. Huang, S. A. Jafar, and J. Li, "Optimal repair of MDS codes in distributed storage via subspace interference alignment," *CoRR*, vol. abs/1106.1250, 2011.
- [22] V. R. Cadambe, S. A. Jafar, and H. Maleki, "Distributed data storage with minimum storage regenerating codes - exact and functional repair are asymptotically equally efficient," *Computing Research Repository (CORR)*, vol. abs/1004.4299, 2010.
- [23] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inform. Theory*, vol. 59, no. 3, pp. 1597–1616, 2013.
- [24] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.
- [25] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inform. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.
- [26] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," *CoRR*, vol. abs/1001.0107, 2010.
- [27] N. Prakash and M. N. Krishnan, "The storage-repair-bandwidth trade-off of exact repair linear regenerating codes for the case $d = k = n - 1$," *arXiv preprint arXiv:1501.03983*, 2015.
- [28] C. Tian, "Rate region of the $(4, 3, 3)$ exact-repair regenerating codes," in *Proc. IEEE Int. Symp. on Infor. Theory (ISIT)*. IEEE, 2013, pp. 1426–1430.
- [29] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth," in *Proc. of the 4th USENIX on Conference on File and Storage Tech. (FAST)*. USENIX Association, 2015.
- [30] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 331–342.
- [31] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.
- [32] C. Tian, V. Aggarwal, and V. A. Vaishampayan, "Exact-repair regenerating codes via layered erasure correction and block designs," *arXiv preprint arXiv:1302.4670*, 2013.
- [33] O. Khan, R. Burns, J. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proc. of USENIX FAST*, 2012.
- [34] "Tahoe-lafs documentation," tahoe-lafs.org, Tech. Rep., 2013. [Online]. Available: <https://tahoe-lafs.org/trac/tahoe-lafs/wiki/Doc>
- [35] J. S. Plank, E. L. Miller, and W. B. Houston, "GF-Complete: A comprehensive open source library for Galois Field arithmetic," University of Tennessee, Tech. Rep. UT-CS-13-703, Jan. 2013.